

Race Cars: Notes On The Second Submission

Cornelia Mayr, Onur Dogangönül

June 20, 2011

Contents

1	Game Description	1
1.1	Controls	2
2	Effects	2
2.1	Shadow Maps	3
2.2	Motion Blur	3
3	Bullet physics	3
3.1	Collision Shapes	3
3.2	Car Physics	3
4	Modelling	4
4.1	Assimp	4
5	Other Technical Aspects	4
5.1	Memory Management	4
5.2	Frustum Culling	4
5.3	Texture Filtering	4
5.4	Lighting	5
5.5	3D Sound	5
6	Goals Until Game Event	5

1 Game Description

Race Cars is a classical racing game intended for two persons, playing in splitscreen mode. The game and racetrack design is inspired by Stunt Car

Racer¹ and Micro Machines². The game takes place in a children's room, the cars are simple toy cars and the track consists of toy bricks. In this setting, which may be considered as "funny", the players are faced with a quite hard challenge of winning the game. The first one who completes the required amount of laps is the winner. Mostly the players are supposed to find a healthy balance between safety and risk: Going too slow will end up in losing the game, going too fast will increase the probability to fall off the track and lose crucial racing time. We are sorry to announce that the collectable items as mentioned in the first design document could not be realized within the scope of this LU.

1.1 Controls

The controls for driving the cars are straight forward, additionally there are keys to reposition the cars. Here is a small summary of the keys needed for playing the game:

- Drive car 1: Arrow keys.
- Drive car 2: W, A, S, D.
- Reposition: (car 1) STRG-RIGHT, (car 2) STRG-LEFT.
- Reposition at last checkpoint: (car 1) SHIFT-RIGHT, (car 2) SHIFT-LEFT.
- Restart current race: ENTF.

To reposition a vehicle, it must come to a halt and is constrained to lie on the back. Currently there are two types of cars available, which may be chosen by pressing UP or DOWN (W, A resp.) in the menu. More details are revealed in the next sections.

2 Effects

Shadow maps were implemented to render shadows, and a very simple form of motion blur was used to increase the dramatic quality while racing.

¹<http://www.youtube.com/watch?v=Kn32IgQGrOQ>

²<http://www.youtube.com/watch?v=EmhtlU8Xv00>

2.1 Shadow Maps

Shadow maps were implemented quite the same way as discussed at the repetitorium, using a FBO with a Frame buffer texture, using hardware PCF and GPU depth comparison.

2.2 Motion Blur

A simple form of motion blur was implemented using PBOs as mentioned in the OpenGL Superbible (Part 2, Chapter 8, Pixel Buffer Objects). The last five frames are saved in a texture ring buffer, and in an additional pass the five frames are summed up to be mapped on to a screen filling rectangle.

3 Bullet physics

The bullet physics library³ is used for the collision detection and physics simulation.

3.1 Collision Shapes

Currently three different collision shapes may be constructed for a given node in the scenegraph.

- BOX: simple but mostly too coarse.
- SIMPLE_CONVEX_HULL: Constructs a simple convex hull, which has less vertices than the detailed convex hull (is used for the car chassis).
- CONCAVE: Constructs a concave collision shape out of convex subparts (used for the track).

There is no collision detection applied for the convex subparts of the track. To render the collision shapes press 1 while playing.

3.2 Car Physics

btRayCastVehicle from bullet was used to simulate the car. Fine tuning is also a goal for the last game event.

³<http://bulletphysics.org/wordpress/>

4 Modelling

4.1 Assimp

Assimp was used to load the models, with textures and material properties. This data is translated to a very basic hierarchical scenegraph which is manipulated by the game and physics layer and rendered by the renderer. The light is also positioned in the scenegraph, geometrically indicated by a transparent cube (look up with the camera).

5 Other Technical Aspects

The game starts with the best looking configuration with trilinear texture filtering, anisotropic filtering, 4x multisampling, shadow maps and motion blur turned on. Also frustum culling, backface culling and transparency is turned on by default. The renderer is a state machine: Press F1 to change the settings.

5.1 Memory Management

The whole data is passed to the VRAM at startup using VBOs and VAOs. There is no implementation using vertex arrays.

5.2 Frustum Culling

Each object is approximated by a sphere and is culled against the view frustum. To see the approximated spheres press 2.

5.3 Texture Filtering

- Nearest neighbor without mipmapping (nn)
- Linear without mipmapping (lin)
- Nearest neighbor with nearest mipmapping.
- Linear with nearest mipmapping (Bilinear).
- Linear with linear mipmapping (Trilinear).

If mipmapping is disabled, texture filtering will toggle between nearest neighbor and linear filtering without mipmapping. Nearest neighbor with

mipmapping, bilinear and trilinear filtering is only available if mipmapping is on. Additionally anisotropic filtering and multisampling is supported.

5.4 Lighting

Mostly a phong shader is used with diffuse, ambient, specular and shininess components. The light is a directional point light source (transparent white node).

5.5 3D Sound

OpenAL and freeAlut were used to implement 3D sound. Player one is always the listener. Every sound node has a world position and velocity.

6 Goals Until Game Event

- Fix motion blur problem on nvidia cards.
- Better shadowmap for the cars.
- Vehicle finetuning.
- Transparency with TGA textures.
- DOF.
- More models.
- Choosing different tracks in the menu.
- Screenshot and video of the final game.